

DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING

COLLEGE OF E&ME, NUST, RAWALPINDI



Project Title: Hospital Emergency Department System Simulation

Course Name: CS212-Object Oriented Programming

SUBMITTED TO:

Ma'am Anum Abdul Salam

SUBMITTED BY:

Names	Registration Numbers
Rida Zahra	465791
Tazeen ur Rehman	466342
Mawa Chaudhary	474121

Submission Date:

29-12-2024

Table of Contents

1.	Introduction
2.	Problem Statement
3.	Objectives
4.	System Design <ul style="list-style-type: none">◦ Class Diagram◦ System Level Diagram
5.	Implementation Details
6.	Key Features
7.	Results and Outputs
8.	Challenges and Solutions
9.	Conclusion
10.	References

1. Introduction

- **Project Overview:**

The Hospital Emergency Department (ED) Simulation for Resource Management project aims to develop a simulation model to optimize patient flow and resource management within a hospital's Emergency Department. The system evaluates the effects of various factors like patient arrival rates, the number of available beds, the acuity of patient conditions, staffing arrangements, and access to inpatient beds. Through this simulation, we can study different scenarios, such as overcrowding or resource allocation, to understand how these changes impact ED performance. Key performance indicators like waiting time, bed utilization, and access block are measured to assess the efficiency of the ED.

- **Hospital Emergency Dependence System and its Importance**

A Hospital Emergency Department (ED) is a critical section of a hospital, providing immediate care to patients in urgent need of medical attention. It serves as the first point of contact for individuals experiencing a variety of health emergencies, ranging from minor injuries to life-threatening conditions. The ED's role is vital in ensuring that patients receive timely medical care, minimizing the risk of complications and optimizing recovery. Given the increasing demand for emergency services and the challenges posed by overcrowding, having an efficient and well-organized ED system is essential for improving patient outcomes and preventing the collapse of healthcare resources.

- **Role of simulations in Healthcare Systems**

Simulations have become a powerful tool in healthcare systems for optimizing operations and managing resources effectively. By simulating the dynamics of emergency departments, hospitals can gain insights into how various factors interact, such as patient arrivals, treatment durations, and staff workload. These simulations help healthcare administrators and medical staff make data-driven decisions to improve the allocation of resources, reduce waiting times, and ensure that patients receive timely care. Furthermore, simulations allow hospitals to model different scenarios—such as peak times or system disruptions—without the risk of affecting real-world operations, making them an indispensable tool in healthcare management.

2. Problem Statement

- The primary problem this project addresses is the simulation of patient flow and resource allocation within a hospital's Emergency Department (ED), focusing on the assignment of patients to triage categories and managing available resources like beds and medical staff. Given the unpredictability of patient arrivals, varying severity levels of medical conditions, and limited resources, the ED faces significant challenges in ensuring timely treatment while minimizing overcrowding and delays. This project aims to model these real-world challenges using simulation techniques to optimize the assignment of resources, reduce waiting times, and improve patient care efficiency.
- The project incorporates the use of probability distribution functions (PDFs) to model key decision-making aspects of the ED system. These include the inter-arrival times

of patients, the triage categories based on the severity of their conditions, and the estimated treatment times. By using probability distributions such as Weibull, Exponential, and Pearson VI, the simulation can account for the randomness in patient arrivals and treatment durations, allowing the ED system to adjust dynamically to changing conditions. These probability functions are crucial for decision-making, helping simulate how different scenarios—such as varying patient inflow or resource availability—impact the performance and effectiveness of the ED.

3. Objectives

- **Simulate ED Patient Flow and Resource Allocation:** Model the movement of patients through the Emergency Department, considering the availability of beds and medical staff to optimize resource allocation and care.
- **Implement an OOP-based, Scalable System:** Use Object-Oriented Programming principles to create a modular and maintainable simulation that can be easily adjusted or scaled for different ED scenarios.
- **Use Probability Distributions for Modeling:** Apply probability functions (e.g., Weibull, Exponential, Pearson VI) to simulate patient arrivals, disease severity, and treatment times, reflecting real-world variability.
- **Evaluate ED Performance with KPIs:** Measure key indicators like waiting times, bed utilization, and access block to assess the efficiency of the ED system and identify areas for improvement.
- **Design and Analyze Optimization Experiments:** Conduct experiments by varying parameters like bed availability and patient arrival rates to find optimal configurations for improving ED performance.

4. System Design

4.1 Classes & Files in the Design:

Classes:

1. Bed: with sub classes :

- Acute Bed
- MinorProcedureRoom
- ReclinerChair
- ResuscitationBed
- Stretchers
- SubacuteBed

2. Physician : with sub classes:

- Junior Resident

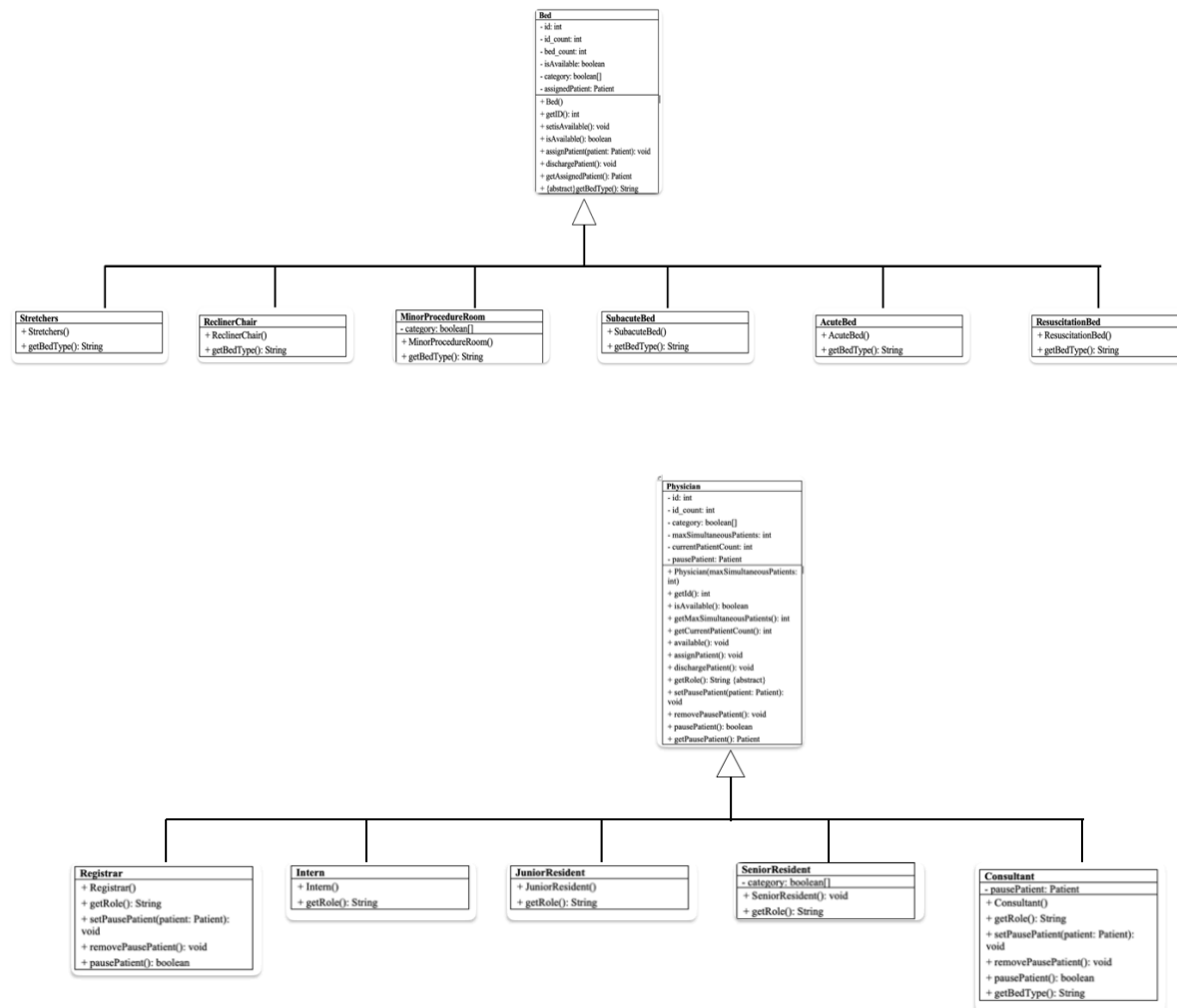
- Registrar
- Senior Resident
- Consultant
- Intern

3. ArrivalTime
4. TreatmentTime
5. TriageCalculator
6. PDDT
7. InpatientBeds
8. InpatientQueue
9. Queue
10. NursingProcedure
11. Laboratory
13. Patient
14. Compiled
15. Simulation (without GUI)
16. Output_GUI (with GUI)

Files:

1. Arrival_output
2. Beds
3. Category_arrival_time_day
4. Category_arrival_time_night
5. Condition_parameters_day
6. Condition_parameters_night
7. Output
8. PDDT_conditions
9. PDDT_results
10. Physician
11. Procedure_times
12. Treatment_capacity
13. Triage_parameters
14. Triage_time
15. Triage_treatmentTimeInp

4.2 Class Diagram



InpatientBeds

- id: int
 - id_count: int
 - bed_count: int
 - isAvailable: boolean
 - assignedPatient: Patient

+ InpatientBeds()
 + getID(): int
 + setisAvailable(): void
 + isAvailable(): boolean
 + assignPatient(patient: Patient): void
 + dischargePatient(): void

+ getAssignedPatient(): Patient + getBedType(): String

InpatientQueue

- queue: List<Patient>

+ InpatientQueue() + arrive(patient: Patient): void + getQueue(): List<Patient> + isEmpty(): boolean + attendNext(): Patient + size(): int + removePatient(patient: Patient): void + display(): void + contains(patient: Patient): boolean
--

ArrivalTime

- PARAMETER_FILE: String - Output_FILE: String

+ ArrivalTime(pf: String, of: String) - weibullPDF(x: double, scale: double, shape: double): double - findMaxDensityTime(scale: double, shape: double, start: double, end: double, step: double): double + simulate(): void
--

Laboratory

- <u>procedure_Capacity</u> : String - capacity: int - currentUsage: int
--

+ Laboratory() + getCapacity(): int + setCapacity(capacity: int) + getCurrentUsage(): int
--

<ul style="list-style-type: none">+ isCapacityFull(): boolean+ addUsage(): boolean+ reduceUsage(): boolean+ readCapacityFromFile(filePath: String)

NursingProcedure

<ul style="list-style-type: none">- capacity: int- currentUsage: int- <u>procedure_Capacity: String</u>
<ul style="list-style-type: none">+ NursingProcedure()+ getCapacity(): int+ setCapacity(capacity: int): void+ getCurrentUsage(): int+ isCapacityFull(): boolean+ addUsage(): boolean+ reduceUsage(): boolean+ readCapacityFromFile(filePath: String): void

Patient

<ul style="list-style-type: none">- id: int- ID_COUNT: int- condition: String- triageCategory: int- arrivalTime: long- TimeofBed: long- TimeofDischarge: long- ArrivaltoInpatientQueue: long- treatmentTime: double- inpatient: boolean- postTreatmentTime: int- assignedBed: Bed- Ibed: InpatientBeds- supervisedBy: Physician
--

- checkedBy: Physician

+ Patient()
+ Patient(condition: String)
+ getId(): int
+ getCondition(): String
+ setCondition(condition: String): void
+ getTriageCategory(): int
+ reduceTriageCategory(): void
+ setTriageCategory(triageCategory: int): void
+ getArrivalTime(): long
+ setArrivalTime(arrivalTime: long): void
+ getInpatientTime(): long
+
setInpatientTime(ArrivaltoInpatientQueue: long): void
+ getTreatmentTime(): double
+ setTreatmentTime(treatmentTime: double): void
+ isInpatient(): boolean
+ setInpatient(inpatient: boolean): void
+ getPostTreatmentTime(): int
+
setPostTreatmentTime(postTreatmentTime: int): void
+ getAssignedBed(): Bed
+ setInpatientBed(Ibed: InpatientBeds): void
+ getInpatientBed(): InpatientBeds
+ setAssignedBed(assignedBed: Bed): void
+ getSupervisedBy(): Physician
+ setSupervisedBy(supervisedBy: Physician): void
+ getCheckedBy(): Physician
+ setCheckedBy(checkedBy: Physician): void
+ getBedID(): int
+ getTimeofBed(): long
+ setTimeofBed(TimeofBed: long): void
+ getTimeofDischarge(): long

+ setTimeofDischarge(TimeofDischarge: long): void + displayDetails(): void + writeDetailsToFile(): void + writeDetailsToFile2(): void + toString(): String

TriageCalculator

- random: Random - <u>PARAMETER_FILE</u> : String
--

+ TriageCalculator() + calculateTriage(conditionName: String): int + readProbabilitiesFromFile(conditionName: String): double[]
--

treatmentTime

- <u>PARAMETER_FILE</u> : String - <u>Output_FILE</u> : String

+ gammaFunction(x: double): double + betaFunction(p: double, q: double): double + pearsonVIDensityFunction(beta: double, p: double, q: double, x: double): double + findMostLikelyTreatmentTime(beta: double, p: double, q: double): double + loadParametersAndProcess(): void

Simulations

+ main(args: String[]): void

Queues

- queue: List<Patient>
+ Queues()

- + arrive(patient: Patient): void
- + getQueue(): List<Patient>
- + isEmpty(): boolean
- + attendNext(): Patient
- + size(): int
- + removePatient(patient: Patient): void
- + display(): void
- + contains(patient: Patient): boolean

Output_GUI

- shell: Shell
- localResourceManager: LocalResourceManager
- text: Text
- comp: Compiled
- text_1: Text
- text_2: Text
- text_3: Text
- simulationRunning: boolean
- lblTotalPatients: Label
- lblTotalPatientIn: Label
- lblTotalPhysicians: Label
- lblTotalBeds: Label
- lblBedsAvailable: Label
- lblTotalPatientIn_1: Label
- lblTimer: Label
- lblTotalPatientIn_2: Label
- lblTotalPatientIn_3: Label
- text_4: Text
- text_5: Text
- text_6: Text
- text_7: Text
- text_9: Text

- + main(args: String[]): void
- + open(): void
- + createContents(): void
- + createResourceManager(): void

+ btnClickToSimulate.mouseDoubleClick(e: MouseEvent): void

PDDT

- <u>PARAMETER_FILE</u> : String

- <u>OUTPUT_FILE</u> : String

+ readConditionsFromFile(): Condition[]

+ calculatePDF(x: double, mean: double): double
--

+ findOptimalPDDT(mean: double): double
--

+ Saving(): void

Compiled

- <u>ARRIVAL_Time_FILE</u> : String

- <u>Triage_Treatment_Time</u> : String

- <u>beds</u> : String

- <u>physicians</u> : String

- <u>procedure_Time</u> : String

- <u>EndTime</u> : long

-currentTime: long

-patients: List<Patient>

-phys: List<Physician>

-mainQueue: Queues

-subQueue: Queues

-procedureQueue: Queues

-laboratoryQueue: Queues

-Inqueue: InpatientQueue

-allBeds: Bed[]

-inpatient: InpatientBeds[]

-bCount: int

-allPhysicians: Physician[]

-triageTreatmentTimes: double[]

-Time_procedure: double[]

-lab: Laboratory

-nursing: NursingProcedure

+Compiled()

+Simm(): void

+triageTreatmentTime(): void

+bed_div(): void

+physician_div(): void

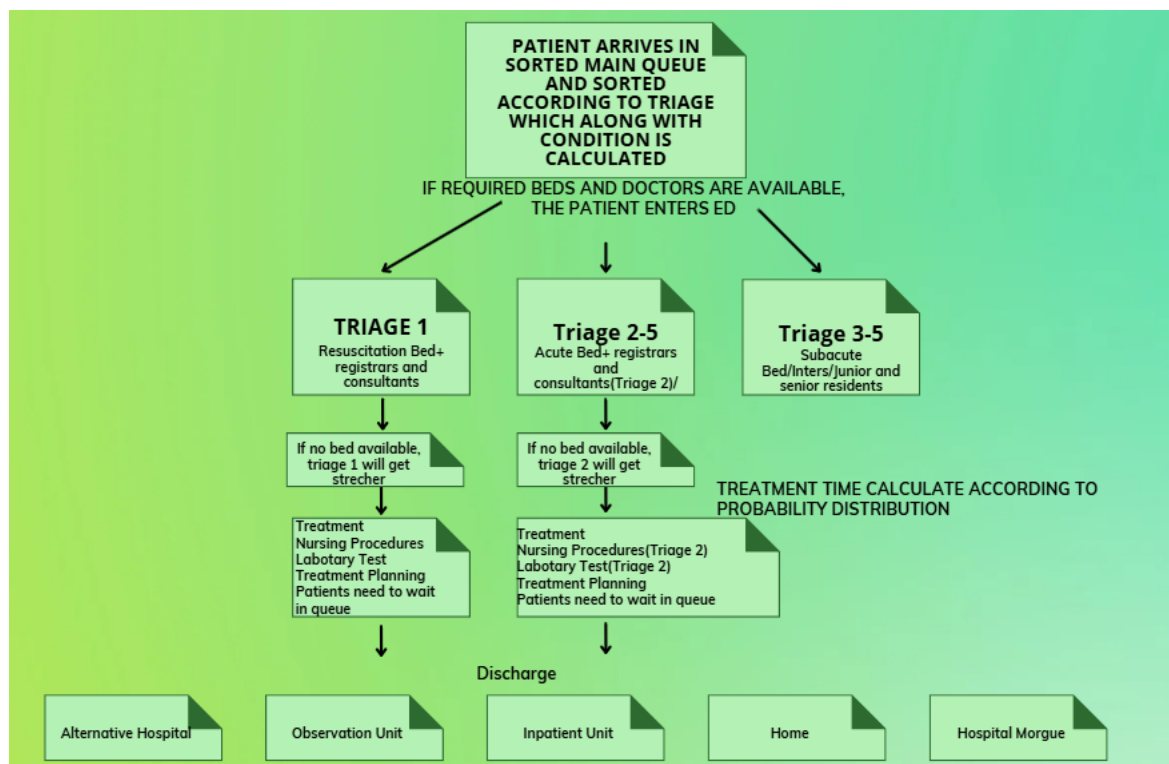
+assignBedToPatient(patient: +Patient): void

```

+assignPatientToPhysician(patient: Patient):
void
+PDDT_check(patient: Patient): void
+processDischarge(patient: Patient): void
+startTreatment(patient: Patient): void
+waiting_time(patient: Patient): void
+setProcedureTimes(): void
+setTime(t: long): void
+NumofPatients(): int
+getMainQueue(): int
+getEDSubQueue(): int
+getprocedureQueue(): int
+getlaboratoryQueue(): int

```

4.2 System Level Diagram



5. Implementation Details

- Programming language used : **JAVA**
- Utilization of OOP Principles
 - **Encapsulation:** Data and behavior related to entities like **Patient** and **Bed** are kept within their respective classes, hiding internal details and promoting data integrity.
 - **Inheritance:** Inheritance allows a class to inherit properties and methods from another class, promoting code reusability and a hierarchical class structure.
 - Physician Classes: The project defines various types of physicians (e.g., Intern, JuniorResident, SeniorResident, Registrar, Consultant) that extend a base class Physician. This structure allows for the definition of common behavior in the Physician class while enabling specific implementations in the subclasses.
 - Bed Classes: Similarly, different types of beds (e.g., SubacuteBed, AcuteBed, MinorProcedureRoom) extend a base class Bed. This design allows shared attributes and methods to be defined in the Bed class, while specific behaviors can be implemented in the subclasses.
- **Polymorphism :** Polymorphism enables methods to perform different tasks based on the object it is acting upon, typically through method overriding or interfaces. The project ensures polymorphism on several instances where required for example methods like **assignPatientToPhysician** method demonstrates polymorphism and method overriding by checking the type of physician (using **instanceof**) to determine which physician can handle a patient based on their triage category. This allows the same method to behave differently depending on the actual object type.
- **Data Persistence through File Handling :** The Project includes mechanisms for reading from and writing to files, demonstrating practical applications of OOP principles for data persistence. For example the code reads patient arrival times from a file (**ARRIVAL_Time_FILE**) using a **BufferedReader**, which allows the program to process each line of data efficiently. Furthermore, after processing the patient data, the project writes relevant patient details to another file using the **writeDetailsToFile2()** method, demonstrating a complete cycle of data persistence. This integration of file handling enhances the simulation's realism and flexibility, allowing it to adapt to varying input conditions while maintaining organized and structured data management.

6. Key Features

Key Features in Detail

Randomized Disease Assignment Based on Triage

The system incorporates a randomized mechanism to assign diseases to patients based on their triage categories. This ensures that patients in higher triage categories (e.g., Category 1 and 2) receive more critical or life-threatening conditions, while patients in lower categories (e.g., Category 3 to 5) receive less severe illnesses. Randomization adds variability and realism, making the simulation dynamic and closely aligned with real-world emergency department scenarios.

Utilization of Object-Oriented Programming (OOP) Principles for Modular Design

The project leverages core OOP principles to ensure that each component is modular, reusable, and easy to maintain:

1. **Encapsulation:** Classes like Patient, Bed, and Physician encapsulate their properties and behaviors, ensuring their internal workings are hidden from other classes. This promotes data integrity and abstraction.
 2. **Inheritance:** A hierarchical structure is used for classes, such as Physician and its subclasses (Intern, JuniorResident, etc.), enabling code reuse and a clear representation of the organizational structure.
 3. **Polymorphism:** Methods like assignPatientToPhysician() and allocateBed() exhibit polymorphic behavior, adapting based on the type of physician or bed, respectively. This ensures flexible and context-specific execution.
-

Extensibility for Adding New Triage Categories or Diseases

The system is designed with future scalability in mind:

1. **Adding New Triage Categories:** The project uses a flexible triage mechanism that allows easy integration of additional categories. The TriageCategory enumeration or equivalent structure can be extended to include new severity levels or custom classifications.
2. **Adding New Diseases:** The disease assignment logic uses a modular design that supports the addition of new conditions. Developers can update a centralized disease repository or add logic to handle new disease types specific to triage levels.

Data Persistence Through File Handling

1. **Input:** Patient arrival times and triage data are read from an input file (ARRIVAL_Time_FILE). This allows the system to simulate various scenarios based on real or pre-defined datasets.
2. **Output:** After processing, the system writes structured details, such as patient assignments, treatment logs, and simulation summaries, to an output file using methods like writeDetailsToFile2(). This ensures a complete cycle of data persistence, enabling record-keeping and further analysis.

Randomized and Dynamic Patient-Physician Interaction

The `assignPatientToPhysician` method utilizes randomization and condition checks to determine the appropriate physician for a patient. Using dynamic properties like `instanceof` and randomized multitasking capabilities, the system ensures:

- **Realistic Load Balancing:** Physicians handle patients according to their capacity (e.g., junior residents seeing 2–3 patients, senior residents managing 2–4 patients).
- **Conditional Supervision:** Interns can treat only with supervision, ensuring realistic constraints.

Comprehensive Bed Management

The project ensures efficient utilization of various types of beds:

- **Subacute Beds:** For less critical cases (e.g., triage Category 3 to 5).
- **Acute Beds:** For critical cases (e.g., triage Category 1 and 2).
- **Minor Procedure Rooms:** For special procedures or treatments.

The bed allocation logic incorporates priority-based assignment, ensuring that critical patients are given priority over less severe cases.

Comprehensive Simulation Workflow

The simulation includes:

1. **Patient Arrival:** Patients arrive randomly or based on input data, ensuring a dynamic flow.
2. **Disease Assignment:** Each patient is assigned a condition based on triage.
3. **Physician Assignment:** Patients are dynamically matched with available physicians.
4. **Bed Allocation:** Beds are allocated based on availability and patient severity.

This workflow ensures end-to-end coverage of emergency department operations.


7. Results and Outputs

The simulation generates detailed results, providing a comprehensive analysis of the emergency department's operations. Key outputs include the number of patients in the main queue, the number being actively treated in the emergency department (ED), and the count of inpatient discharges during the simulation run. The total number of physicians, available ED beds, and inpatient beds are also summarized. Patient-specific details, such as arrival time, triage category, assigned physician, and discharge status, are recorded and saved to a file for further analysis. This output allows users to assess the efficiency of the system under varying conditions and evaluate the resource allocation strategies in real time, supporting improvements in emergency care planning.

The results emphasize the system's behavior during peak hours, particularly at night when patient arrivals and severity are highest. Increasing the number of available beds significantly reduces patient wait times and queue lengths, enabling smoother operations and improved resource utilization. With more beds, the ED can accommodate higher patient volumes during nighttime peaks, reducing bottlenecks and enhancing care delivery. Conversely, limited bed availability results in longer queues, higher physician workloads, and delayed treatments, highlighting the critical role of capacity planning in emergency care management.

```
Arrival_output - Notepad
File Edit Format View Help
ID: 465791
Patient Condition: Gastrointestinal
Triage Category: 5
Arrival Time: 50
Inpatient: No
Treatment Time: 41.0
-----
ID: 465792
Patient Condition: Multitrauma
Triage Category: 1
Arrival Time: 55
Inpatient: Yes
Arrival to Inpatient Queue Time: 0
Post-Treatment Time: 462
Treatment Time: 31.0
-----
ID: 465793
Patient Condition: GP referred /Hosp transfer
Triage Category: 2
Arrival Time: 55
Inpatient: No
Treatment Time: 25.0
-----
ID: 465794
Patient Condition: Pain
Triage Category: 3
Arrival Time: 70
Inpatient: No
Treatment Time: 39.0
-----
ID: 465795
Patient Condition: Diabetes/Endocrine
Triage Category: 3
Arrival Time: 80
Inpatient: No
Treatment Time: 39.0
-----
ID: 465796
Patient Condition: Paediatric
Triage Category: 3
Arrival Time: 80
Inpatient: No
Treatment Time: 39.0
<
```

File about Patient's details as soon as it enters the main Queue

 output - Notepad

File Edit Format View Help

ID: 465793

Patient Condition: GP referred /Hosp transfer

Triage Category: 2

Arrival Time: 55

Time of Bed: 55

Inpatient: No

Treatment Time: 50.0

Time of Discharge: 115

ID: 465791

Patient Condition: Gastrointestinal

Triage Category: 5

Arrival Time: 50

Time of Bed: 50

Inpatient: No

Treatment Time: 82.0

Time of Discharge: 140

ID: 465798

Patient Condition: Eye

Triage Category: 2

Arrival Time: 85

Time of Bed: 85

Inpatient: No

Treatment Time: 50.0

Time of Discharge: 145

ID: 465805

Patient Condition: Renal

Triage Category: 2

Arrival Time: 120

Time of Bed: 120

Inpatient: No

Treatment Time: 50.0

Time of Discharge: 180

File about the Patient as it gets discharged

SWT Application

Welcome to MTR Simulation

Enter Time of Simulation

[Click to Simulate](#)

SWT Application

Welcome to MTR Simulation

Enter Time of Simulation

[Click to Simulate](#)

Total Patients	<input type="text" value="41"/>	Total Physicians	<input type="text" value="120"/>
Total Patient in Main Queue	<input type="text" value="23"/>	Total Beds	<input type="text" value="116"/>
Total Patients Inpatient	<input type="text" value="4"/>	Total Inpatient Beds	<input type="text" value="50"/>
Patients Discharged	<input type="text" value="10"/>		

Timer

SWT Application

Welcome to MTR Simulation

Enter Time of Simulation

[Click to Simulate](#)

Total Patients	<input type="text" value="94"/>	Total Physicians	<input type="text" value="120"/>
Total Patient in Main Queue	<input type="text" value="53"/>	Total Beds	<input type="text" value="116"/>
Total Patients Inpatient	<input type="text" value="14"/>	Total Inpatient Beds	<input type="text" value="50"/>
Patients Discharged	<input type="text" value="25"/>		

Timer

8. Challenges and Solutions

Challenges Faced During Development

1. **Implementing Probability Distributions:** One of the significant challenges encountered was the implementation of probability distributions to simulate patient arrival times and treatment durations. Accurately modeling these distributions was crucial for creating a realistic simulation of an emergency department, as it directly impacts patient flow and resource allocation.
2. **Debugging Complex Logic:** The project involved intricate logic for managing patient triage, bed assignments, and physician availability. Debugging this complex logic proved to be challenging, especially when trying to ensure that patients were assigned to the correct resources based on their triage categories and that the simulation accurately reflected real-world scenarios.
3. **File Handling Issues:** Managing file input and output presented its own set of challenges. Ensuring that the program could read from and write to files without errors required careful handling of file formats and data parsing. Additionally, handling exceptions related to file operations was essential to prevent the program from crashing due to unexpected input.
4. **Concurrency Management:** The simulation required real-time processing of patient data, which involved managing multiple threads for patient treatment and resource allocation. Ensuring that shared resources (like beds and physicians) were accessed safely without causing race conditions was a significant challenge.
5. **Queue Management and Patient Prioritization:** A significant challenge faced during the development of the project was effectively managing patient queues and ensuring that patients were prioritized correctly based on their triage categories. The emergency department simulation required a dynamic queue system that could handle varying patient arrival rates and prioritize treatment based on the severity of conditions. Ensuring that patients with higher triage categories received timely attention while managing the flow of lower-priority patients was complex and required careful consideration of queue operations.

Solutions Implemented

1. **Utilizing Libraries for Probability Distributions:** To address the challenge of implementing probability distributions, the project leveraged existing libraries (such as Apache Commons Math) that provide built-in functions for generating random numbers based on various statistical distributions. This approach simplified the implementation process and ensured that the distributions were accurate and reliable.
2. **Incremental Debugging and Logging:** To tackle the complex logic and debugging challenges, an incremental development approach was adopted. The code was developed and tested in smaller modules, allowing for easier identification of issues. Additionally, extensive logging was implemented throughout the code to track the flow of data and the state of objects, making it easier to pinpoint errors and understand the behavior of the simulation.
3. **Robust File Handling Mechanisms:** To resolve file handling issues, the code was structured to include comprehensive error handling using try-catch blocks. This ensured that any exceptions related to file operations were caught and managed gracefully, allowing the program to continue running or provide informative error messages. Input validation was also implemented to ensure that the data read from files was in the expected format before processing.

9. Conclusion

- **Achievements:**
 - Developed a simulation for managing Emergency Department (ED) patient flow and resource allocation.
 - Implemented an object-oriented approach for a modular and scalable system.
 - Used probability distributions to model patient arrivals, triage decisions, and treatment durations.
 - Provided valuable insights on ED performance, such as treatment times, resource utilization, and waiting times.
- **Potential for Further Development:**
 - Integrating real-time patient data for dynamic simulation and more accurate modeling.
 - Incorporating machine learning to optimize staffing, bed allocation, and patient treatment predictions.
 - Enhancing predictive capabilities for improving hospital efficiency and patient care.

10. References

Research Paper : Hospital Emergency Department Simulation for Resource Analysis Erhan Kozan† and Mel Diefenbach, School of Mathematical Sciences Queensland University of Technology

Online Resources : Chatgpt, YouTube